Decision problems in membrane systems with peripheral proteins, transport and evolution

Technical Report CoSBi 12/2006

Decision Problems in Membrane Systems with Peripheral Proteins, Transport and Evolution

Matteo Cavaliere

The Microsoft Research – University of Trento Centre for Computational and Systems Biology

Sean Sedwards

The Microsoft Research – University of Trento Centre for Computational and Systems Biology

This is a preliminary version of a paper that will appear in

Decision Problems in membrane systems with peripheral proteins, transport and evolution, Theoretical Computer Science, 404, pp. 40–51, 2008.

Available from

`http://www.elsevier.com/locate/tcs`

Abstract

Transport of substances and communication between compartments are fundamental biological processes, often mediated by the presence of complementary proteins attached to the surfaces of membranes. Within compartments, substances are acted upon by local biochemical rules. Inspired by this behaviour we present a model based on Membrane Systems, with objects attached to the sides of the membranes and floating objects that can be moved between the regions of the system. Moreover, in each region there are evolution rules that *rewrite* the transported objects, mimicking chemical reactions.

We investigate qualitative properties, like configuration reachability, in relation to the use of cooperative or non-cooperative evolution and transport rules and in the contexts of free- and maximal-parallel evolution.

1 Introduction and Motivations

Membrane Systems (also known as P systems) are models of computation inspired by the structure and function of biological cells. The model was introduced in 1998 by Gh. Păun and since then many results have been obtained, mostly concerning computational power (for an updated bibliography the reader can consult the web-page [23]). More recently, membrane systems have been applied to systems biology and several models have been proposed for simulating biological processes (e.g., see the monograph dedicated to membrane systems applications, [9]).

In the original definition, membrane systems are composed of an hierarchical nesting of membranes that enclose regions in which floating objects exist. Each region can have associated rules for evolving these objects (called evolution rules, modelling the biochemical reactions present in cell regions), and/or rules for moving objects across membranes (called symport/antiport rules, modelling some kinds of transport mechanisms present in cells). Recently, inspired by *brane calculus* [4], a model of a membrane system, having objects attached to the membranes, was introduced in [5]. Other models bridging brane calculus and membrane systems have been proposed in [14, 17]. A more general approach, considering both free floating objects and objects attached to the membranes has been proposed and investigated in [3]. The idea of these models is that membrane operations are moderated by the objects (proteins) attached to the membranes. However, in all these models objects are associated to an atomic membrane which has no concept of inner or outer surface. In reality, many biological processes are driven and controlled by the presence of specific proteins on the appropriate sides of a membrane. For instance, endocytosis, exocytosis and budding in cells are processes where the existence and locality of membrane proteins is crucial (see, e.g., [1]).

In general, the compartments of a cell are in constant communication, with molecules being passed from a donor compartment to a target compartment, mediated by membrane proteins. Once transported to the correct compartment the substances are often then *processed* by means of local biochemical reactions.

Motivated by this, we investigate a model combining some basic features found in biological cells: (*i*) *evolution* of objects (molecules) by means of multiset rewriting rules

associated with specific regions of the systems (the rules model biochemical reactions); (*ii*) *transport* of objects across the regions of the system by means of rules associated with the membranes of the system and involving proteins attached to the membranes (on one or possibly both sides) and (*iii*) rules that take care of the *attachment/de-attachment* of objects to/from the sides of the membranes. Moreover, since we want to distinguish the functioning of different regions, we also associate to each membrane a unique identifier (a label).

In this paper we present a qualitative investigation of the model using two alternative evolution strategies. The first is based on *free parallelism*: at each step of the evolution of the system an arbitrary number of rules may be applied. We prove that, in this case, useful properties like configuration reachability can be decided, even in the presence of cooperative evolution and transport rules.

We also consider *maximal parallel* evolution: if a rule can be applied it *must* be applied, with alternative possible rules being chosen non-deterministically. This strategy models, for example, the behaviour in biology where a process takes place as soon as resources become available. In this case we show that configuration reachability becomes an undecidable property when the systems use non-cooperative evolution rules coupled with cooperative transport rules. However, several other cases where the problem remains decidable are also presented.

We wish to comment that the model presented follows the philosophy of the evolution-communication model introduced in [6], where the system evolves by evolution of the objects and transport of objects by means of symport/antiport rules that are essentially synchronized exchanges of objects. However, in the model presented here the transport of objects may depend on the presence of particular *proteins* attached to the internal and external surfaces of the membranes. Therefore this paper can be seen as a bridge between membrane systems and *projective brane calculus* [10], where, in the framework of process algebra, directed actions associated to membranes have been considered.

The paper is an extension of the work present in [7].

2 Formal Language Preliminaries

We will briefly recall the main notions and results of the formal language theory used in this paper. For more details the reader can consult standard books, such as [13], [22], [11], and the respective chapters of the handbook [21].

Given a set A, we denote by $|A|$ its cardinality. The empty set is denoted by $\emptyset$.

As usual, an *alphabet* V is a finite set of symbols. By V^* we denote the set of all strings over V. The empty string is denoted by λ.

The *length* of a string $w \in V^*$ is denoted by $|w|$, while the number of occurrences of $a \in V$ in w is denoted by $|w|_a$.

Given an alphabet $V = \{a_1, a_2, \ldots, a_n\}$, for all strings $x \in V^*$ we can associate the *Parikh vector* $\Psi_V(x) = (|x|_{a_1}, |x|_{a_2}, \ldots, |x|_{a_n})$. Given a language $L \subseteq V^*$, we can also define the *Parikh image* of L as $\Psi_V(L) = \{\Psi_V(x) \mid x \in L\}$.

The notation $Perm(x)$ indicates the set of all strings that can be obtained as a permutation of the string x.

For $x, y \in V^*$ we define their *shuffle* by $x\xi y = \{x_1y_1 \cdots x_ny_n \mid x = x_1 \cdots x_n,$
$y = y_1 \cdots y_n, x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1\}$. The operation can be extended in an intuitive way to languages. Then, given L_1 and L_2, we have $L_1\xi L_2 = \bigcup_{x_1 \in L_1, x_2 \in L_2} x_1\xi x_2$.

Denoting by REG the family of regular languages, the following result holds (see, e.g., [21]) (proved in a constructive way).

Theorem 2.1 *$L_1, L_2 \in REG$, then $L_1\xi L_2 \in REG$.*

A *multiset* over a set V is a map $M : V \to \mathbb{N}$, where $M(a)$ denotes the multiplicity of the symbol $a \in V$ in the multiset M. This fact can also be indicated by the forms $(a, M(a))$ or $a^{M(a)}$, for all $a \in V$. If the set V is finite, e.g. $V = \{a_1, \ldots, a_n\}$, then the multiset M can be explicitly described as $\{(a_1, M(a_1)), (a_2, M(a_2)), \ldots, (a_n, M(a_n))\}$. The *support* of a multiset M is the set $supp(M) = \{a \in V \mid M(a) > 0\}$. A multiset is empty (so finite) when its support is empty (also finite).

A compact notation can be used for finite multisets: if $M = \{(a_1, M(a_1)), (a_2, M(a_2)),$ $\ldots, (a_n, M(a_n))\}$ is a multiset of finite support, then the string $w = a_1^{M(a_1)}a_2^{M(a_2)} \ldots a_n^{M(a_n)}$ (and all its possible permutations) precisely identify the symbols in M and their multiplicities. Hence, given a string $w \in V^*$, we can assume that it identifies a finite multiset over V defined by $M(w) = \{(a, |w|_a) \mid a \in V\}$.

In this paper we make use of the notion of a *matrix grammar*.

A *matrix grammar with appearance checking (ac)* is a construct
$G = (N, T, S, M, F)$, where N and T are disjoint alphabets of non-terminal and terminal symbols, $S \in N$ is the axiom, M is a finite set of matrices which are sequences of context-free rules of the form $(A_1 \to x_1, \ldots, A_n \to x_n), n \geq 1$ (with $A_i \in N, x_i \in (N \cup T)^*$ in all cases), and F is a set of occurrences of rules in M.

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either
(i) $w_i = w_i'A_iw_i'', w_{i+1} = w_i'x_iw_i''$, for some $w_i', w_i'' \in (N \cup T)^*$
or
(ii) $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F.

The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied (one says that these rules are applied in *appearance checking* mode).

The family of languages generated by matrix grammars with appearance checking is denoted by MAT_{ac}.

G is called a *matrix grammar without appearance checking* if and only if $F = \emptyset$. In this case the generated family of languages is denoted by MAT.

If we denote by CF and RE the family of context-free and recursively enumerable languages, respectively, then the following results hold:

Theorem 2.2

- $CF \subset MAT \subset RE$.

- $MAT \subset MAT_{ac} = RE$.

A matrix grammar is called *pure* if there is no distinction between terminals and non-terminals. The language generated by a pure matrix grammar is composed of all the sentential forms. The family of languages generated by pure matrix grammars without appearance checking is denoted by $pMAT$. A proof of this can be found, for example, in [11].

Theorem 2.3 $pMAT \subset MAT$

A context-free *programmed grammar* with appearance checking is a construct $G = (N, T, S, P)$, where N, T, S are the set of non-terminals, the set of terminals and the start symbol, respectively, and P is a finite set of rules of the form $(b : A \to x, E_b, F_b)$, where b is a label, $A \to x$ is a context-free rule over $N \cup T$, and E_b, F_b are two sets of labels of rules of G (E_b is called the *success field* and F_b the *failure field* of the rule). If the failure field is empty for any rule of P, then the grammar is without appearance checking. We denote $Lab(P) = \{b \mid (b : A \to x, E_b, F_b) \in P\}$.

The language $L(G)$ generated by G is defined as the set of all the words $w \in T^*$ such that there is a derivation

$$S = w_0 \Rightarrow_{b_1} w_1 \Rightarrow_{b_2} w_2 \Rightarrow_{b_3} \cdots \Rightarrow_{b_k} w_k = w,$$

$k \geq 1$, and, for $(b_i : A_i \to x_i, E_{b_i}, F_{b_i})$, $1 \leq i \leq k$, one of the following conditions hold: $w_{i-1} = w'_{i-1} A_i w''_{i-1}, w_i = w'_{i-1} x_i w''_{i-1}$ for some $w'_{i-1}, w''_{i-1} \in (N \cup T)^*$ and $b_{i+1} \in E_{b_i}$ or A_i does not occur in w_{i-1}, $w_{i-1} = w_i$ and $b_{i+1} \in F_{b_i}$.

In other words, a rule $(b_i : A_i \to x_i, E_{b_i}, F_{b_i})$ is applied as follows: if A_i is present in the sentential form, the rule is used and the next rule to be applied is chosen from those with the label in E_{b_i}, otherwise the sentential form remains unchanged and we choose the next rule from the rules labelled by some element of F_{b_i} and try to apply it. Without loss of generality we suppose that there is a unique initial production having the axiom S called the *initial production* of G.

By PR we denote the family of languages generated by programmed grammars without appearance checking and by PR_{ac} we denote the family of languages generated by programmed grammars with appearance checking.

The following theorem is true (see, e.g., [11]).

Theorem 2.4 $MAT = PR \subset MAT_{ac} = PR_{ac} = RE$.

The literature is rich with parallel rewriting devices, where the rewriting of the current sentential form is performed in a parallel way and not sequentially (as in the previously described grammars). Lindenmayer systems (or *L systems* for short) are possibly the most well known parallel rewriting systems.

An ET0L system is a construct $G = (\Sigma, T, H, w)$, where Σ is the alphabet, $T \subseteq \Sigma$ is the terminal alphabet; $H = \{h_1, h_2, \cdots, h_k\}$ is a finite set of finite substitutions (tables) over Σ and $w \in \Sigma^*$ is the axiom; each $h_i \in H$, $1 \leq i \leq k$, can be represented by a list of

context-free productions $A \to x$, such that $A \in \Sigma$ and $x \in \Sigma^*$ (moreover, for each symbol A of Σ and each table h_i, $1 \leq i \leq k$, there is a production in h_i with A as left hand side); G defines a derivation relation $\Rightarrow_{h_i}$ by $x \Rightarrow_{h_i} y$ iff $y \in h_i(x)$, for some $1 \leq i \leq k$ (h_i is used as substitution). We write only $x \Rightarrow y$ if we are not interested in the table used.

The language generated by G is $L(G) = \{z \in T^* \mid w \Longrightarrow^* z\}$. We denote by $ET0L$ the family of languages generated by ET0L systems.
In what follows we assume the reader to be familiar with the basic notions of membrane systems, for instance, as presented in the introductory guide [20].

3 Membrane Operations with Peripheral Proteins

As is usual in the membrane systems field, a membrane is represented by a pair of square brackets, [ ]. To each topological *side* of a membrane we associate multisets u and v (over a particular alphabet V) and this is denoted by $[\,_u]_v$. We say that the membrane is *marked* by u and v; v is called the *external marking* and u the *internal marking*; in general, we refer to them as *markings* of the membrane. The objects of the alphabet V are called *proteins* or, simply, *objects*. An object is called *free* if it is not attached to the sides of a membrane, so is not part of a marking.

Each membrane encloses a *region* and the *contents* of a region can consist of free objects and/or other membranes (we also say that the region *contains* free objects and/or other membranes).

Moreover, each membrane has an associated label that is written as a superscript of the square brackets. If a membrane is associated to the label i we call it membrane i. Each membrane encloses a unique region, so we also say region i to identify the region enclosed by membrane i. The *set of all labels* is denoted by Lab.

For instance, in the system $[\, abb\, [\, aaaa\,_{ab}]^2_{b\ bba}]^1_{ab}$, the external membrane, labelled by 1, is marked by bba (internal marking) and by ab (external marking). The contents of the region enclosed by the external membrane is composed of the free objects a, b, b and the membrane $[\, aaaa\,_{ab}]^2_b$.

We consider rules that model the attachment of objects to the sides of the membranes. These rules extend the definition given in [3].

$$attach:\ [\, a\,_u]^i_v \to [\,_{ua}]^i_v, \quad a[\,_u]^i_v \to [\,_u]^i_{va}$$
$$de-attach:\ [\,_{ua}]^i_v \to [a\,_u]^i_v, \quad [\,_u]^i_{va} \to [\,_u]^i_v a$$

with $a \in V$, $u, v \in V^*$ and $i \in Lab$.

The semantics of the attachment rules (*attach*) is as follows.

For the first case, the rule is *applicable* to the membrane i if the membrane is marked by multisets *containing* the multisets u and v on the appropriate sides, and region i contains an object a. In the second case, the rule is applicable to membrane i if it is marked by multisets containing the multisets u and v, as before, and is contained in a region that contains an object a. If the rule is applicable we say that the objects defined by u, v and a can be *assigned* to the rule (so that it may be executed).

In both cases, if a rule is applicable and the objects given in u, v and a are assigned to the rule, then the rule can be executed and the object a is added to the appropriate marking in the way specified. The objects not involved in the application of a rule are left unchanged in their original positions.

The semantics of the de-attachment rule (*de-attach*) is similar, with the difference that the attached object a is detached from the specified marking and added to the contents of either the internal or external region.

We now consider rules associated to the membranes that control the passage of objects across the membranes:

$$move_{in}: \ a[\ _u]_v^i \rightarrow [\ a\ _u]_v^i$$
$$move_{out}: \ [\ a\ _u]_v^i \rightarrow a[\ _u]_v^i$$

with $a \in V$, $u, v \in V^*$ and $i \in Lab$.

The semantics of the rules is as follows.

In the first case, the rule is applicable to membrane i if it is marked by multisets containing the multisets u and v, on the appropriate sides, and the membrane is contained in a region containing an object a. The objects defined by u, v and a can thus be assigned to the rule.

If the rule is applicable and the objects a, u and v are assigned to the rule then the rule can be executed and, in this case, the object a is removed from the contents of the region surrounding membrane i and added to the contents of region i.

In the second case the semantics is similar, but here the object a is moved from region i to its surrounding region.

The rules of attach, de-attach, move$_{in}$, move$_{out}$ are generally called *membrane rules* (denoted collectively as mem_{rul}) over the alphabet V and the set of labels Lab.

Membrane rules for which $|uv| \geq 2$ we call *cooperative* membrane rules (in short, coo_m). Membrane rules for which $|uv| = 1$ are called *non-cooperative* membrane rules (in short, $ncoo_m$). Membrane rules for which $|uv| = 0$ are called *simple* membrane rules (in short, sim_m).

We also introduce *evolution rules* that involve objects but not membranes. These can be considered to model the biochemical reactions that take place inside the compartments of the cell. They are evolution rules over the alphabet V and set of labels Lab and they follow the definition that can be found in evolution-communication P systems [6]. We define

$$evol: \ [u \rightarrow v]^i$$

with $u \in V^+, v \in V^*$ and $i \in Lab$. An evolution rule is called *cooperative* (in short, coo_e) if $|u| > 1$, otherwise the rule is called *non-cooperative* ($ncoo_e$).

The rule is applicable to region i if the region *contains* a multiset of free objects that *includes* the multiset u. The objects defined by u can thus be assigned to the rule.

If the rule is applicable and the objects defined by u are assigned to the rule, then the rule can be executed. In this case the objects specified by u are subtracted from the contents of region i while the objects specified by v are added to the contents of the region i.

4 Membrane Systems with Peripheral Proteins

In this section we define membrane systems having membranes marked with multisets of proteins on both sides of the membrane, free objects and using the operations introduced in Section 3.

Formally, a *membrane system with peripheral proteins* (in short, a P_{pp} system) and n membranes, is a construct

$$\Pi = (V, \mu, (u_1, v_1), \ldots, (u_n, v_n), w_1, \ldots, w_n, R, R^m)$$

- V is a finite, non-empty alphabet of objects (proteins).
- μ is a membrane structure with $n \geq 1$ membranes, injectively labelled by $1, 2, \ldots, n$.
- $(u_1, v_1), \cdots, (u_n, v_n) \in V^* \times V^*$ are the markings associated, at the beginning of any evolution, to the membranes $1, 2, \cdots, n$, respectively. They are called *initial markings* of Π; the first element of each pair specifies the internal marking, while the second one specifies the external marking.
- $w_1, \cdots, w_n$ specify the multisets of free objects contained in regions $1, 2, \cdots$ $\cdots, n$, respectively, at the beginning of any evolution and they are called *initial contents* of the regions.
- R is a finite set of evolution rules over V and the set of labels $Lab = \{1, \ldots, n\}$.
- R^m is finite set of membrane rules over the alphabet V and set of labels $Lab = \{1, \ldots, n\}$.

5 Evolution of the System

A *configuration* of Π consists of a membrane structure, the markings of the membranes (internal and external) and the multisets of free objects present inside the regions. In what follows, configurations are denoted by writing the markings as subscripts (internal and external) of the parentheses which identify the membranes. The labels of the membranes are written as superscripts and the contents of the regions as string, e.g.,

$[\,[\,{}_{aa}]^4_{ab}\,[aaa\,{}_{aa}]^2_b\,[\,b\,]^3_{bb}\,a\,]^1_a$

We suppose a standard labelling: 0 is the label of the *environment* that surrounds the entire system Π; 1 is the label of the *skin* membrane that separates Π from the *environment*.

The *initial configuration* consists of the membrane structure μ, the initial markings of the membranes and the initial contents of the regions; the environment is empty at the beginning of the evolution.

We denote by $\mathbb{C}(\Pi)$ the set of all possible configurations of Π.

We assume the existence of a clock which marks the timing of steps (single *transitions*) for the whole system.

A transition from a configuration C to a new one is obtained by assigning the objects present in the configuration to the rules of the system and then executing the rules as described in Section 3.

We define two possible ways of assigning the objects to the rules: free-parallel and maximal-parallel.

- *Free-Parallel Evolution.*

  In each region and for each marking, *an arbitrary number* of applicable rules is executed (membrane and evolution rules have equal precedence). A single object (free or not) may only be assigned to a single rule.

  This implies that in one step, no rule, one rule or as many applicable rules as desired may be applied. This strategy is similar to one introduced in ([19], Chapter 3.4).

  We call a single transition performed in a free-parallel way a *free-parallel transition.*

- *Maximal-Parallel Evolution.*

  In each region and for each marking, applicable rules chosen in a non-deterministic way are assigned objects, also chosen in a non-deterministic way, such that after the assignment no further rule is applicable for the unassigned objects. As with free-parallel evolution, membrane and evolution rules have equal precedence and a single object (free or not) may only be assigned to a single rule.

  We call a single transition performed in a maximal-parallel way a *maximal-parallel transition.*

A sequence of free-parallel [maximal-parallel] transitions, starting from the initial configuration, is called a free-parallel [maximal-parallel, resp.] *evolution.* An evolution (free or maximal parallel) is said to be *halting* if it halts, that is, if it reaches a *halting configuration*, i.e., a configuration where no rule can be applied anywhere in the system.

A configuration of a P_{pp} system Π that can be reached by a free-parallel [maximal-parallel] evolution, starting from the initial configuration, is called free-parallel [maximal-parallel, resp.] *reachable.* A pair of multisets (u, v) is a free-parallel [maximal-parallel] *reachable marking* for Π if there exists a free-parallel [maximal-parallel, resp.] reachable configuration of Π which contains at least one membrane marked internally by u and externally by v.

We denote by $\mathbb{C}_R(\Pi, fp)$ $[\mathbb{C}_R(\Pi, mp)]$ the set of all free-parallel [maximal parallel, resp.] reachable configurations of Π and by $\mathbb{M}_R(\Pi, fp)$ $[\mathbb{M}_R(\Pi, mp)]$ the set of all free-parallel [maximal-parallel, resp.] reachable markings of Π.

Moreover, we denote by $\mathbb{P}_{pp,m}(\alpha, \beta)$, $\alpha \in \{coo_e, ncoo_e\}$, $\beta \in \{coo_m, ncoo_m, sim_m\}$ the class of membrane systems with peripheral proteins, evolution rules of type α, membrane rules of type β, and m membranes (m is changed to $*$ if it is unbounded). We omit α or β from the notation if the corresponding types of rules are not allowed. We also denote by V_Π the alphabet V of the system Π.

6 Reachability with Free-Parallel Evolution

We would like to know whether or not a biological system can evolve to a particular specified configuration. Hence it would be useful to construct models having such qualitative properties to be decidable.

Using our model we can prove that when the evolution is free-parallel it is possible to decide, for an arbitrary membrane system with peripheral proteins and an arbitrary configuration, whether or not such a configuration is reachable by the system. A proof can be demonstrated by showing that all the reachable configurations of a system Π can be produced by a pure matrix grammar without appearance checking. Moreover, we also prove that the reachability of an arbitrary marking can be decided.

Lemma 6.1 *It is decidable whether or not, for any P_{pp} system Π from $\mathbb{P}_{pp,1}(coo_e)$ and any configuration C of Π, $C \in \mathbb{C}_R(\Pi, fp)$.*

Proof Let $\Pi = (V, \mu = [\ ]^1, (u_1, v_1), w_1, R)$. We first notice that since membrane rules are excluded, any configuration C of Π is effectively the contents of the unique region and therefore, being a multiset, can be represented by a string w_C, as described in Section 2 (every permutation of the string w_C represents the same contents, so the same configuration C). We construct a pure matrix grammar G without appearance checking such that $L(G)$ contains all and only the strings representing the configurations in $\mathbb{C}_R(\Pi)$.

The grammar $G = (N, S, M)$ is defined in the following way. $N = V \cup V^\#$, with $V^\# = \{v^\# \mid v \in V\}$. We add to M the matrix $(S \to w_1)$ and, for each rule $[x \to y]^1 \in R$, the matrix

$$(x_1 \to x_1^\#, x_2 \to x_2^\#, \cdots, x_k \to x_k^\#, x_1^\# \to \lambda, x_2^\# \to \lambda, \cdots, x_k^\# \to y_1 y_2 \cdots y_q)$$

where $x = x_1 x_2 \cdots x_k$ and $y = y_1 y_2 \cdots y_q$. Each application of a matrix simulates the application of an evolution rule inside the unique region of the system. The markings are not involved in the evolution of the system since membrane rules are not allowed. We can see immediately that, for each string w in $L(G)$ (i.e., all the sentential forms generated by G) there is an evolution of Π, starting from the initial configuration, that reaches the configuration represented by w. Moreover, it is easy to see that the reverse is also true since the evolution of Π is based on free parallelism: for each reachable configuration C' of Π there exists a derivation of G that generates a string representing C'. In fact it can be seen that $L(G)$ contains all the strings representing configurations of Π reached by applying at each step a single evolution rule. In the case a configuration C' is reached by applying more than a unique evolution rule in a single step, a single step can be simulated in G by applying an appropriate sequence of matrices.

Therefore, to check whether or not an arbitrary configuration C of Π can be reached, we only need to check if any of the strings representing C is in $L(G)$. This can be done since there is only a finite number of strings representing C and the membership problem for pure matrix grammars without appearance checking is decidable (for the proof see [12]); therefore the Lemma follows. □

Theorem 6.1 *It is decidable whether or not, for any P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_e, coo_m)$ and any configuration C of Π, $C \in \mathbb{C}_R(\Pi, fp)$.*

Proof The main idea of the proof is that the problem can be reduced to check whether or not a configuration of a system from $\mathbb{P}_{pp,1}(coo_e)$ is reachable, and this is decidable (Lemma 6.1).

Suppose $\Pi = (V, \mu, (u_1, v_1), \ldots, (u_n, v_n), w_1, \ldots, w_n, R, R^m)$. By $cont(i)$ we denote the label of the region surrounding membrane i (we recall that 0 is the label of the environment and 1 is the label of the skin membrane).

We construct $\overline{\Pi} = (\overline{V}, [\ ]^1, (\lambda, \lambda), \overline{w_1}, \overline{R})$ from $\mathbb{P}_{pp,1}(coo_e)$ in the following way.

We define $\overline{V} = \bigcup_{i\in\{1,\cdots,n\}}(V_i' \cup V_i'') \cup \bigcup_{i\in\{0,1,\cdots,n\}} V_i$ with $V_i = \{a_i \mid a \in V\}$, $V_i' = \{a_i' \mid a \in V\}$, $V_i'' = \{a_i'' \mid a \in V\}$.

We use the morphisms h_i, h_i', h_i'', defined as follows.

- $h_i : V \to V_i$ defined by $h_i(a) = a_i, a \in V$, for $i \in \{0, 1, \cdots, n\}$
- $h_i' : V \to V_i'$ defined by $h_i'(a) = a_i', a \in V$, for $i \in \{1, \cdots, n\}$
- $h_i'' : V \to V_i''$ defined by $h_i''(a) = a_i'', a \in V$, for $i \in \{1, \cdots, n\}$

We define $\overline{w_1}$ as the string $h_1(w_1)\cdots h_n(w_n)h_1'(u_1)\cdots h_n'(u_n)h_1''(v_1)\cdots h_n''(v_n)$.

For each rule $move_{in}$, $a[\ _u]_v^i \to [\ a\ _u]_v^i \in R^m$, $i \in \{1, \cdots, n\}$ we add to $\overline{R}$ the following rules: $[\ a_k h_i'(u)h_i''(v) \to a_i h_i'(u)h_i''(v)]^1$, with $k = cont(i)$.

In the same way all the other rules present in $R \cup R^m$ can be *translated* in the evolution rules for $\overline{R}$.

Hence, given a configuration C of Π, one can construct the configuration $\overline{C}$ of $\overline{\Pi}$ having a unique region in the following way.

For each free object a contained in region i (the environment if $i = 0$) in C, $i \in \{0, 1, \cdots, n\}$ we add the object $h_i(a)$ in region 1 of $\overline{C}$. For each object a present in the internal marking of membrane i in C, $i \in \{1, \cdots, n\}$ we add the object $h_i'(a)$ to region 1 of $\overline{C}$ and finally for each object a present in the external marking of membrane i, $i \in \{1, \cdots, n\}$ we add the object $h_i''(a)$ to region 1 of $\overline{C}$.

Now we can decide (Lemma 6.1) whether or not $\overline{C} \in \mathbb{C}_R(\overline{\Pi})$.

From the way $\overline{\Pi}$ has been constructed it follows that:

- if $\overline{C} \in \mathbb{C}_R(\overline{\Pi})$ then $C \in \mathbb{C}_R(\Pi)$.
- if $\overline{C} \notin \mathbb{C}_R(\overline{\Pi})$ then $C \notin \mathbb{C}_R(\Pi)$.

and from this the Theorem follows.

□

Remark: Notice that in the proof of Theorem 6.1, the reduction of the problem of deciding whether or not, for any P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_e, coo_m)$ and any configuration C of Π, $C \in \mathbb{C}_R(\Pi, fp)$ to the corresponding problem for matrix grammars without a.c., has been obtained by a *polynomial-time reduction* [16] (with respect to the size of the system Π and configuration C).

Without going into details, we also sketch the proof of the "reverse" Theorem.

Theorem 6.2 *For any pure matrix grammar $G = (N, S, M)$ without a.c. there exists a P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_e)$ (constructed by using a polynomial-time reduction with respect to G) such that, given an arbitrary string $w \in N^*$, $w \in L(G)$ if and only if $C_w \in \mathbb{C}_R(\Pi, fp)$ with C_w a configuration of Π obtained from w by a polynomial-time reduction (with respect to w).*

Proof Given a pure matrix grammar $G = (N, S, M)$. Suppose, without loss of generality, that M has n matrices (indicated by $m_i, 1 \leq i \leq n$) and each matrix has p productions. So $m_{i,k}$, $1 \leq i \leq n, 1 \leq k \leq p$ indicates the production p of matrix i.

We then construct Π in the following way.

$$\Pi = (V, [\ ]_1, (\lambda, \lambda), w_1, R = \emptyset, R^{ev})$$

with $V = N \cup \{(i,k) \mid 1 \leq i \leq n, 1 \leq k \leq p\}$. For each matrix $m_l : (1 : A_1 \rightarrow \alpha_1, 2 : A_2 \rightarrow \alpha_2, \cdots, p : A_p \rightarrow \alpha_p), 1 \leq l \leq n$ we add to R^{ev} the evolution rules $[(l,1)A_1 \rightarrow \alpha_1(l,2)]^1$, $[(l,2)A_2 \rightarrow \alpha_2(l,3)]^1, \cdots, [(l,2)A_p \rightarrow \alpha_p(i,1)]^1$, $1 \leq i \leq n$.

From the construction we have that an arbitrary $w \in N^*$ is in $L(G)$ if and only if C_w is in $\mathbb{C}_R(\Pi, fp)$, where C_w is the configuration of Π represented by (any of) the strings (finite in number) $w \cdot (i,k), 1 \leq i \leq n, 1 \leq k \leq p$ (the string $w \cdot (i,k)$ represents the multiset of objects contained in region 1 in configuration C_w).

□

Corollary 6.2.a *It is decidable whether or not, for any P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_e, coo_m)$ and any pair of multisets (u, v) over V_Π, $(u, v) \in \mathbb{M}_R(\Pi, fp)$.*

Proof Given Π from $\mathbb{P}_{pp,n}(mem_{rul}, coo_e)$ and with alphabet of objects V, one can construct $\overline{\Pi} = (\overline{V}, \mu = [\ ]^1, (\lambda, \lambda), \overline{w_1}, \overline{R})$ from $\mathbb{P}_{pp,1}(coo_e)$ in the way described by Theorem 6.1.

Therefore, using $\overline{\Pi}$ one can construct the grammar G as described by Lemma 6.1 such that $L(G)$ contains all and only the strings representing the configurations in $\mathbb{C}_R(\overline{\Pi}, fp)$.

Now, to check whether or not an arbitrary $(u, v) \in \mathbb{M}_R(\Pi, fp)$ one needs to check whether or not there exists an $i \in \{1, \cdots, n\}$ such that
$(Perm(h'_i(u))\xi(\overline{V})^*) \cap L(G) \neq \emptyset$ *and* $(Perm(h''_i(v))\xi(\overline{V})^*) \cap L(G) \neq \emptyset$, where h'_i and h''_i are morphisms from V to V'_i and to V''_i, respectively, defined as in Theorem 6.1, and ξ denotes the shuffle operation.

The permutation and shuffle operation are used to construct all possible strings representing a configuration of $\overline{\Pi}$ containing the membrane i marked by multiset u internally and by multiset v externally.

The languages $(Perm(h_i'(u))\xi(\overline{V})^*) \cap L(G)$ and $(Perm(h_i''(v))\xi(\overline{V})^*) \cap L(G)$ can be generated by matrix grammars without appearance checking (see Theorem 2.1 and e.g., [11]) and the emptiness problem for this class of grammars is decidable (see, e.g., [11]). Therefore the Corollary follows. □

7 Reachability with Maximal-Parallel Evolution

If the system evolves in a maximal-parallel way then we prove decidability when the evolution rules used are non-cooperative and the membrane rules are simple or when the system uses only membrane rules (including cooperative membrane rules).

We further show that it is undecidable whether or not an arbitrary configuration can be reached by an arbitrary system working in the maximal-parallel way and using non-cooperative evolution rules coupled with cooperative membrane rules. The proof is based on the fact that, in this case, a P_{pp} system can simulate the derivations of a programmed grammar with appearance checking.

We first analyse systems with only membrane rules.

Theorem 7.1 *It is decidable whether or not:*

- *For an arbitrary P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_m)$ and an arbitrary configuration C of Π, $C \in \mathbb{C}_R(\Pi, mp)$.*

- *For an arbitrary P_{pp} system Π from $\mathbb{P}_{pp,*}(coo_m)$ and an arbitrary pair of multisets u, v over V_Π, $(u, v) \in \mathbb{M}_R(\Pi, mp)$.*

Proof Given a P_{pp} system from $\mathbb{P}_{pp,*}(coo_m)$ the number of possible reachable configurations for Π is finite because the system can only use membrane rules (which neither add nor remove objects). So the problem is decidable (by an exhaustive search). □

We now investigate systems having non-cooperative evolution and simple membrane rules.

Lemma 7.1 *It is decidable whether or not, for an arbitrary P_{pp} system Π from $\mathbb{P}_{pp,1}(ncoo_e)$ and an arbitrary configuration C of Π, $C \in \mathbb{C}_R(\Pi, mp)$.*

Proof

Let $\Pi = (V, \mu = [\ ]^1, (u_1, v_1), w_1, R)$. As already mentioned in Lemma 6.1, any configuration C of Π is effectively the contents of the unique region and therefore, being a multiset, can be represented by a string w_C (every permutation of the string w_C represents

the same contents, so the same configuration C). We construct an $ET0L$ system $G = (\Sigma, \Sigma, h_1, w_1)$ (i.e., only one table and $\Sigma = T$) such that $L(G)$ contains all and only the strings representing the configurations in $\mathbb{C}_R(\Pi, mp)$.

The grammar $G = (\Sigma, \Sigma, h_1, w_1)$ is defined in the following way. $\Sigma = V$. We add to h_1 the production $(S \rightarrow w_1)$ and, for each rule $[a \rightarrow \alpha]^1 \in R$, the production $a \rightarrow \alpha$.

The markings are not involved in the evolution of the system since membrane rules are not allowed. It is immediately clear that for each string w in $L(G)$ (i.e., all the sentential forms generated by G) there is an evolution of Π, starting from the initial configuration, that reaches the configuration represented by w. Moreover, it is easy to see that, for each reachable configuration C of Π, there exists a derivation of G that generates a string representing C (because Π works in maximal parallel way).

Therefore to check whether or not an arbitrary configuration C of Π can be reached, we only need to check if any of the strings representing C is in $L(G)$. This can be done since there is only a finite number of strings representing C and the membership problem for $ET0L$ systems is decidable (see, e.g., [11]); therefore the Lemma follows.

□

Theorem 7.2 *It is decidable whether or not, for an arbitrary P_{pp} system Π from $\mathbb{P}_{pp,*}(ncoo_e, sim_m)$ and an arbitrary configuration C of Π, $C \in \mathbb{C}_R(\Pi, mp)$.*

Proof

The idea of the proof closely follows the one given in 6.1, so we only give a sketch here.

Suppose $\Pi = (V, \mu, (u_1, v_1), \ldots, (u_n, v_n), w_1, \ldots, w_n, R, R^m)$.

We construct $\overline{\Pi} = (\overline{V}, [\,]^1, (\lambda, \lambda), \overline{w_1}, \overline{R})$ from $\mathbb{P}_{pp,1}(ncoo_e)$ using the morphisms h_i, h_i', h_i'', as in Theorem 6.1. In this way it is easy to see that (using the same idea of Theorem 6.1), given an arbitrary configuration of Π, $C \in \mathbb{C}_R(\Pi, mp)$ if and only if $C \in \mathbb{C}_R(\overline{\Pi}, mp)$.

The Theorem follows using Lemma 7.1.

□

Corollary 7.2.a *It is decidable whether or not, for any P_{pp} system Π from $\mathbb{P}_{pp,*}(ncoo_e, sim_m)$ and any pair of multisets (u, v) over V_Π, $(u, v) \in \mathbb{M}_R(\Pi, mp)$.*

Proof The idea of the proof follows closely the one given in Corollary 6.2.a so once again we only give a sketch.

Suppose $\Pi = (V, \mu, (u_1, v_1), \ldots, (u_n, v_n), w_1, \ldots, w_n, R, R^m)$. We construct $\overline{\Pi} = (\overline{V}, [\,]^1, (\lambda, \lambda), \overline{w_1}, \overline{R})$ from $\mathbb{P}_{pp,1}(ncoo_e)$ using the morphisms h_i, h_i', h_i'', as in Theorem 6.1. Using $\overline{\Pi}$ one can construct an ET0L system G as described by Lemma 7.1 such that $L(G)$ contains all and only the strings representing the configurations in $\mathbb{C}_R(\overline{\Pi}, mp)$.

Now, to check whether or not an arbitrary pair of multisets over V_Π (u, v) is in $\mathbb{M}_R(\Pi, mp)$ one needs to check whether or not there exists an $i \in \{1, \cdots, n\}$ such that $(Perm(h_i'(u))\xi(\overline{V})^*) \cap L(G) \neq \emptyset$ *and* $(Perm(h_i''(v))\xi(\overline{V})^*) \cap L(G) \neq \emptyset$ (ξ denotes the shuffle operation).

The permutation and shuffle operation are used to construct all possible strings representing a configuration of $\overline{\Pi}$ containing the membrane i marked by multiset u internally and by multiset v externally.

The languages $(Perm(h_i'(u))\xi(\overline{V})^*) \cap L(G)$ and $(Perm(h_i''(v))\xi(\overline{V})^*) \cap L(G)$ can be generated by an ET0L system (see Theorem 2.1 and e.g., [11]) and the emptiness problem for ET0L systems is decidable (see, e.g., [11]). Therefore the Corollary follows.

□

We investigate now systems having non-cooperative evolution rules and cooperative membrane rules, showing that in this case the reachability of an arbitrary configuration becomes an undecidable problem.

Theorem 7.3 *It is undecidable whether or not, for an arbitrary P_{pp} system Π from $\mathbb{P}_{pp,*}(ncoo_e, coo_m)$ and an arbitrary configuration C of Π, $C \in \mathbb{C}_R(\Pi, mp)$.*

Proof Given a programmed grammar $G = (N, T, S, P)$ with appearance checking, as defined in Section 2, suppose that $Lab(P) = \{0, 1, 2, \cdots, n\}$ and 0 is the label of the initial production of G. We denote by $\overline{N} = \{\overline{x} \mid x \in N\}$ and by $\overline{T} = \{\overline{a} \mid a \in T\}$. We use the morphism $h : N \cup T \to \overline{N} \cup \overline{T}$ defined by $h(x) = \overline{x}$ for $x \in N \cup T$. We indicate by A_i the non-terminal on the left-hand side of the production with label i.

We construct the following P_{pp} system Π defined as:

$$\Pi = (V, \mu, (u_1, v_1), (u_2, v_2), (u_3, v_3), w_1, w_2, w_3, R, R^{ev})$$

with

$$\begin{aligned}
V = \;& N \cup T \cup \overline{N} \cup \overline{T} \cup V' \cup V'' \text{ with} \\
V' = \;& \{l_i, l_i', l_i'', \overline{l_i}, \overline{\overline{l_i}} \mid i \in Lab(P)\} \cup \{\#\} \cup \{Y', Y'', \cdots, Y^{v\iota\iota}, d, l_{-1}\} \\
V'' = \;& \{h_i^s, \overline{h_i^s}, \overline{l_i^s}, \overline{\overline{l_i^s}}, \overline{\overline{\overline{l_i^s}}}, (l_i^s)' \mid i \in Lab(P)\} \\
\cup \;& \{X^s, (X^s)', (X^s)'', (X^s)''', (X^s)^{iv}, Y^s, (Y^s)', (Y^s)'', \cdots, (Y^s)^{v\iota\iota\iota}\} \\
\mu = \;& [\,[\,]^2\,[\,]^3\,]^1 \\
& u_1 = v_1 = u_2 = v_2 = u_3 = v_3 = \lambda \\
& w_2 = \lambda, w_3 = \lambda, w_1 = l_{-1} S h_1^s \cdots h_n^s \\
R = \;& R' \cup R'' \text{ with} \\
R' = \;& \{l_i'[\,]^2 \to [\,]^2_{l_i'} \mid i \in Lab(P)\} && (1) \\
\cup \;& \{A[\,]^2_{l_i'} \to [A\,]^2_{l_i'} \mid i \in Lab(P), A \in N\} && (2) \\
\cup \;& \{[\overline{x}]^2 \to [\,]^2 \overline{x} \mid x \in N \cup T\} && (3) \\
\cup \;& \{[\overline{l_i}]^2 \to [\,]^2 \overline{l_i},\ [\,]^2_{l_i'} \to [\,]^2 l_i',\ l_i''[\,]^3 \to [\,]^3_{l_i''} \mid i \in Lab(P)\} && (4) \\
\cup \;& \{\overline{\overline{l_i}}[\,]^3_{l_i''} \to [\overline{\overline{l_i}}\,]^3_{l_i''}, [\,\overline{\overline{l_i}}]^3_{l_i''} \to [\,\overline{\overline{\overline{l_i}}}]^3_{l_i''},\ \mid i \in Lab(P)\} && (5) \\
\cup \;& \{Y^{v\iota\iota}[\,\overline{\overline{\overline{l_i}}}]^3_{l_i''} \to [Y^{v\iota\iota}\,\overline{\overline{\overline{l_i}}}]^3_{l_i''},\ [\,\overline{\overline{\overline{l_i}}}]^3_{l_i''} \to [\,\overline{\overline{\overline{l_i}}}]^3 l_i'', [\,\overline{\overline{\overline{l_i}}}]^3 \to [\,\overline{\overline{l_i}}]^3 \mid i \in Lab(P)\} && (6)
\end{aligned}$$

$$
\begin{aligned}
R'' = \quad & \{(l_i^s)'[\,]^2 \to [\,]^2_{(l_i^s)'},\ h_i^s[\,]^2_{(l_i^s)'} \to [\,h_i^s]^2_{(l_i^s)'}, [\,]^2_{(l_i^s)'} \to [\,]^2(l_i^s)' \mid i \in Lab(P)\} && (7)\\
\cup \quad & \{[\,\overline{l_i^s}]^2 \to [\,]^2\overline{l_i^s},\ [\,\overline{h_i^s}]^2 \to [\,]^2\overline{h_i^s} \mid i \in Lab(P)\} && (8)\\
\cup \quad & \{(l_i^s)'[\,]^3 \to [\,]^3_{(l_i^s)'},\ (X^s)^{iv}[\,]^3_{(l_i^s)'} \to [\,(X^s)^{iv}]^3_{(l_i^s)'} \mid i \in Lab(P)\} && (9)\\
\cup \quad & \{A[\,]^3_{(l_i^s)'} \to [\,A]^3_{(l_i^s)'} \mid i \in Lab(P), A \in N\} && (10)\\
\cup \quad & \{\overline{\overline{l_i^s}}[\,]^3_{(l_i^s)'} \to [\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'},\ [\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'} \to [\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'} \mid i \in Lab(P)\} && (11)\\
\cup \quad & \{(Y^s)^{v\iota\iota\iota}[\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'} \to [(Y^s)^{v\iota\iota\iota}\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'} \mid i \in Lab(P)\} \cup \{l_{-1} \to l_0'Y\} && (12)\\
\cup \quad & \{[\,\overline{\overline{l_i^s}}]^3_{(l_i^s)'} \to [\,\overline{\overline{l_i^s}}]^3(l_i^s)',\ [\,\overline{\overline{l_i^s}}]^3 \to [\overline{\overline{l_i^s}}]^3 \mid i \in Lab(P)\} && (13)\\
R^{ev} = \quad & (R^{ev})' \cup (R^{ev})'' \text{ with} &&\\
(R^{ev})' = \quad & \{[l_j \to l_i'Y]^1 \mid i \in E(j), j \in Lab(P)\} \cup \{[l_{-1} \to l_0'Y]^1\} && (14)\\
\cup \quad & \{[Y \to Y']^1,\ [Y' \to Y'']^1, \cdots, [Y^{v\iota} \to Y^{v\iota\iota}]^1,\ [Y^{v\iota\iota} \to \#]^1\} && (15)\\
\cup \quad & \{[\overline{x} \to x]^1 \mid x \in N \cup T\} && (16)\\
\cup \quad & \{[\overline{l_i} \to \overline{\overline{l_i}}]^1 \mid i \in Lab(P)\} && (17)\\
\cup \quad & \{[l_i' \to l_i'']^1,\ [l_i'' \to l_i]^1 \mid i \in Lab(P)\} && (18)\\
\cup \quad & \{[A \to h(\alpha)\overline{l_i}]^2 \mid (i : A \to \alpha, E(i), F(i)) \in P\} && (19)\\
\cup \quad & \{[Y^{v\iota\iota} \to \lambda]^3\} && (20)\\
\cup \quad & \{[\overline{\overline{l_i}} \to d]^3 \mid i \in Lab(P)\} && (21)\\
\cup \quad & \{[d \to \lambda]^3\} && (22)\\
(R^{ev})'' = \quad & \{[l_j \to (l_i^s)'Y^sX^s]^1 \mid i \in E(j), j \in Lab(P)\} && (23)\\
\cup \quad & \{[l_j^s \to l_i'Y]^1 \mid i \in F(j), j \in Lab(P)\} && (24)\\
\cup \quad & \{[l_j^s \to (l_i^s)'Y^sX^s]^1 \mid i \in F(j), j \in Lab(P)\} && (25)\\
\cup \quad & \{[X^s \to (X^s)']^1,\ [(X^s)' \to (X^s)'']^1, [(X^s)'' \to (X^s)''']^1, && (26)\\
& [(X^s)''' \to (X^s)^{iv}]^1, [(X^s)^{iv} \to \#]^1\} && (27)\\
\cup \quad & \{[\overline{h_i^s} \to h_i^s]^1, [\overline{l_i^s} \to \overline{\overline{l_i^s}}]^1,\ [\overline{\overline{l_i^s}} \to \overline{\overline{\overline{l_i^s}}}]^1,\ [\overline{\overline{\overline{l_i^s}}} \to \#]^1, [(l_i^s)' \to l_i^s]^1 \mid i \in Lab(P)\} && (28)\\
\cup \quad & \{[(Y^s) \to (Y^s)']^1,\ [(Y^s)' \to (Y^s)'']^1,\ [(Y^s)'' \to (Y^s)''']^1, && (29)\\
& [(Y^s)''' \to (Y^s)^{iv}]^1, [(Y^s)^{iv} \to (Y^s)^{v}]^1,\ [(Y^s)^{v} \to (Y^s)^{v\iota}]^1, && (30)\\
& [(Y^s)^{v\iota} \to (Y^s)^{v\iota\iota}]^1,\ [(Y^s)^{v\iota\iota} \to (Y^s)^{v\iota\iota\iota}]^1, [(Y^s)^{v\iota\iota\iota} \to \#]^1\} && (31)\\
\cup \quad & \{[h_i^s \to \overline{h_i^s}\ \overline{l_i^s}]^2,\ [\overline{\overline{\overline{l_i^s}}} \to d]^3, \mid i \in Lab(P) && (32)\\
\cup \quad & \{[A \to \#]^3 \mid A \in N\} \cup \{[(Y^s)^{v\iota\iota\iota} \to \lambda]^3\}. && (33)
\end{aligned}
$$

Note that where each numbered line contains a list of rules, the first in the list will be referred to in the text as *number*.a, the second as *number*.b etc.

The basic idea of the proof is that the system Π simulates the derivations of the grammar G, storing in region 2 a multiset of objects corresponding to the current sentential form of the grammar. In this way a reachability problem in G can be reduced to a reachability problem in Π and so, since programmed grammars with a.c. have been

proved universal (in a constructive way, see Section 2) then the theorem holds.

We have divided the alphabet, the evolution rules and the transport rules into subsets. V', R' and $(R^{ev})'$ are used during the simulation of the application of production of G while V'', R'' and $(R^{ev})''$ are used for the simulation of the skipping of a production of G (the appearance checking case). We use the objects (present in region 1) $l_i, i \in Lab(P)$ to indicate the label (i) of the last simulated production, in case it was applied, and objects $l_i^s, i \in Lab(P)$ to indicate the label of the last simulated production (i) in case it was skipped.

We show in detail the functioning of Π.

Suppose that the last simulated production has label j and *it has been applied* (the case where the last simulated production has been skipped is similar).

Then, at some step $t-1$, the object l_j is present in region 1, together with the objects corresponding to the current sentential form of G and the objects $h_i^s, i \in Lab(P)$.

Region 2 and 3 as well as the markings are empty. As particular case we have the initial configuration, where $l_j = l_{-1}$, the only applicable next rule is 14.b. However, in general, the next rule of Π to apply is chosen (in a non-deterministic way) from rules in groups 14.a and 23.

We distinguish two cases.

- **Case 1**

A rule $[l_j \to l_i' Y]^1$ for some $i \in E(j)$ is applied at step t.

The application of such a rule means that Π has "guessed" that the next production of G that has to be simulated and that *can actually be applied* is the one with label i. The application of this rule produces two objects l_i' and Y.

(i) Suppose that, at step $t+1$, the object l_i' attaches to membrane 2 using the rule $l_i'[\,]^2 \to [\,]^2_{l_i'}$. In the same step the object Y is rewritten to Y' (rule 15.a).

(ii) Suppose that at step $t+2$ an object A present in region 1 (corresponding to the non-terminal A in N) is introduced to region 2 using one of the rules of group 2. In the same step Y' is rewritten to Y''.

At step $t+3$ the object A is rewritten inside region 2 using one of the rules in group 19.

(iii) Suppose the rule used is $A \to h(\alpha)\overline{l_k}$ with $k = i$ (so $\overline{l_k} = \overline{l_i}$).

(iv) In the same step $t+3$, object l_i' is de-attached from membrane 2 using a rule from 4.b and Y'' is rewritten to Y'''.

At step $t+4$ the objects of $h(\alpha)$ and $\overline{l_i}$ move from region 2 to region 1 (rules from group 3 and 4.a, resp.), while l_i' is rewritten to l_i'' (rule 18.a).

In the same step Y''' is rewritten to Y^{iv}.

In step $t+5$ the objects from $h(\alpha)$ are rewritten to α (the bar is removed) (rules 16); the multiset of objects in region 1 corresponding to the current sentential form of G is updated as the production i of G has been applied. Moreover, $\overline{l_i}$ is rewritten to $\overline{\overline{l_i}}$ (rules 17), l_i'' attaches to membrane 3 using the rule in 4.3 (it is the only rule that can use this object). In the same step, the object Y^{iv} is rewritten to Y^v.

In step $t+6$ the object $\overline{\overline{l_i}}$ moves from region 1 to region 3 using the object l_i'' on membrane 3 and rule 5.a.

In the same step the object Y^v is rewritten to $Y^{v\iota}$.

In step $t+7$, $Y^{v\iota}$ becomes $Y^{v\iota\iota}$ while $\overline{\overline{l_i}}$ is attached (internally) to membrane 3 using rule 5.a.

In step $t+8$, the object $Y^{v\iota\iota}$ moves from region 1 to region 3 using the rule $Y^{v\iota\iota}[\ _{\overline{\overline{l_i}}}]^3_{l_i''} \rightarrow [Y^{v\iota\iota}\ _{\overline{\overline{l_i}}}]^3_{l_i''}$ from group 6.a.

In step $t+9$, $Y^{v\iota\iota}$ is deleted in region 3, while l_i'' de-attaches from membrane 3 using the rule 6.a.

It is possible for l_i'' to attach/de-attach to/from membrane 3 using, an arbitrary number of times, the rules from group 4.c and 6.b, resp. At a certain step $t+9+p$, l_i'' is rewritten to l_i in region 1 (rule 18.b). This is necessary to start a new simulation of a production of G.

Moreover, at step $q \leq t+9+p+1$, $\overline{\overline{l_i}}$ de-attaches from membrane 3 and goes into region 3 (rule 6.c) and is then rewritten to d at step $q+2$ and then deleted at step $q+2$ ($\overline{\overline{l_i}}$ cannot attach back to membrane since it would need l_i'' that is missing).

In this way, the production i of G with $i \in E(j)$ has been correctly simulated (in particular, applied) and at the step $t+9+p+1$ a new rule among the rules in 14.a and 23. is applied and so the entire process can be iterated.

We now discuss the assumptions made during the described evolution of Π and we show that if the assumptions are not true then # is eventually produced in region 1 (notice that there are no rules to remove #).

For assumptions (i), (ii) & (iv):

When l_i' is produced (step t), Y is also produced and is ultimately rewritten to $Y^{v\iota\iota}$ at step $t+7$.

If l_i' is not attached to membrane 2 at step $t+1$ (and hence rewritten to l_i'') or it is de-attached from membrane 3 before an object A is transported from region 1 to region 2 (meaning it is de-attached from membrane 2 at step $t+2$ or A is not present in region 1) then the rule $A \rightarrow h(\alpha)\overline{l_i}$ is not used in region 2 at step $t+2$, and so $\overline{l_i}$ is not produced at step $t+3$ and then it cannot be attached to membrane 3 at step $t+7$. So, at step $t+8$, the rule $Y^{v\iota\iota}[\ _{\overline{\overline{l_i}}}]^3_{l_i''} \rightarrow [Y^{v\iota\iota}\ _{\overline{\overline{l_i}}}]^3_{l_i''}$ cannot be used. Therefore, rule $Y^{v\iota\iota} \rightarrow \#$ is used and # is produced in region 1.

On the other hand, if l_i'' is not obtained (from l_i') in region 1 at step $t+4$ then l_i'' cannot be attached to membrane 3 at step $t+5$ (so $\overline{\overline{l_i}}$ cannot be attached to membrane 3 at step $t+7$) and then Y cannot be moved inside region 3 at step $t+8$. Therefore $Y^{v\iota\iota} \rightarrow \#$ is used and # is produced in region 1.

Hence, to avoid creation of # in region 1, l_i' must attach to membrane 2 at step $t+1$, must de-attach from it at step $t+3$ and be rewritten to l_i'' at step $t+4$.

Assumption (iii):

If the rule used is $A \rightarrow h(\alpha)\overline{l_k}$ with $k \neq i$ then at step $t+8$ the rule $Y^{v\iota\iota}[\ _{\overline{\overline{l_i}}}]^3_{l_i''} \rightarrow [Y^{v\iota\iota}\ _{\overline{\overline{l_i}}}]^3_{l_i''}$ cannot be used ($\overline{\overline{l_i}}$ is not attached to membrane 3) and so $Y^{v\iota\iota} \rightarrow \#$ is used in

region 1.

Consider now the second case.

- **Case 2: appearance checking**

A rule $[l_j \to (l_i^s)'Y^sX^s]^1$ for some $i \in E(j)$ is applied at step t. The application of this rule means that Π has "guessed" that the next production of G that has to be simulated and that *should be skipped because it cannot be applied* is the one with label i. The application of this rule produces the objects $(l_i^s)'$, Y^s and X^s.

(i) At step $t+1$ the object $(l_i^s)'$ attaches to membrane 2 using a rule of group 7.a.

In the same step X^s is rewritten to $(X^s)'$ and Y^s is rewritten to $(Y^s)'$.

(ii) In step $t+2$, the object h_i^s moves from region 1 to region 2 using a rule of group 7.b. In the same step objects $(X^s)'$ and $(Y^s)'$ are rewritten to $(X^s)''$ and $(Y^s)''$ respectively.

In step $t+3$, object h_i^s, in region 2, is rewritten to $\overline{h_i^s}$ $\overline{l_i^s}$ using rule 32.a. In the same step $(l_i^s)'$ de-attaches from membrane 2 using rule 7.c (it is the only rule that can involve the object). Also, the objects $(X^s)''$ and $(Y^s)''$ are rewritten to $(X^s)'''$ and $(Y^s)'''$ respectively.

In step $t+4$, objects $\overline{h_i^s}$ and $\overline{l_i^s}$ move from region 2 to region 1 using rules 8.a and 8.b.

(iii) In the same step object $(l_i^s)'$ attaches to membrane 3 using rule 9.a.

Moreover, objects $(X^s)'''$ and $(Y^s)'''$ are rewritten to $(X^s)^{iv}$ and $(Y^s)^{iv}$, respectively.

In step $t+5$, object $(X^s)^{iv}$ move from region 1 to region 3 using rule 9.b. In the same step $(Y^s)^{iv}$ is rewritten to $(Y^s)^v$. Moreover, in region 1, $\overline{h_i^s}$ is rewritten to h_i^s using rule 28.a and $\overline{l_i^s}$ is rewritten to $\overline{\overline{l_i^s}}$ using rule 28.b.

(iv) Suppose that, at step $t+6$, there is no object A_i in region 1. Then, in this step, $(Y^s)^v$ is rewritten to $(Y^s)^{v\iota}$ and $\overline{\overline{l_i^s}}$ is rewritten to $\overline{\overline{\overline{l_i^s}}}$ using rule 28.c.

In step $t+7$, $\overline{\overline{\overline{l_i^s}}}$ moves from region 1 to region 3 using rule 11.a while $(Y^s)^{v\iota}$ is rewritten to $(Y^s)^{v\iota\iota}$.

In step $t+8$, $\overline{\overline{\overline{l_i^s}}}$ attaches (internally) to membrane 3 using rule 11.b. Moreover, $(Y^s)^{v\iota\iota}$ is rewritten to $(Y^s)^{v\iota\iota\iota}$.

In step $t+9$, the object $(Y^s)^{v\iota\iota\iota}$ moves from region 1 to region 3 using rule 12.a.

In step $t+10$, the object $(Y^s)^{v\iota\iota\iota}$ is deleted inside region 3.

For an arbitrary number of steps the objects $(l_i^s)'$ and $\overline{\overline{\overline{l_i^s}}}$ can iterate their attachment/de-attachment to/from membrane 3 using rules 13.a, 9.a or 13.b and 11.b. However, to start a new simulation of a production of G the object $(l_i^s)'$ needs to be de-attached from membrane 3 (step $t+10+p$) and then rewritten (step $t+10+p+1$) to l_i^s using rule 28.e. So, at step $t+10+p+2$ the object l_i^s is obtained in region 1. The object indicates that the last simulated (and skipped) production of G is the one with label i.

Moreover, at step $q \leq t+10+p+1$ object $\overline{\overline{\overline{l_i^s}}}$ must de-attach from membrane 3 using 13.b (there are no other rules) and then rewritten to d (step $q+1$) inside region 3 using rule 32.b (there are no other rules available; l_i^s is not available anymore on membrane 3). Finally d is deleted (step $q+2$ using rule 22.a).

In this way, the production i of G with $i \in E(j)$ has been correctly simulated (in particular, skipped) and at the step $t + 10 + p + 2$ the process can then be iterated by choosing, in a non-deterministic way, one of the rules in 24.a or 25.a (then case 1 or case 2 can be applied again).

We now discuss the assumptions made during the description of the process and we show that if the assumptions are not true then # is produced in region 1.

Assumption (i): Suppose that at step $t + 1$ the object $(l_i^s)'$ does not attach to membrane 2 (but chooses another possible rule). Then, in this case, object h_i^s cannot move from region 1 to region 2 at step $t+2$. Then it is not possible to fulfill both the conditions:

$(l_i^s)'$ attached to membrane 3 at step $t+4$ (to let $(X^s)^{iv}$ move from region 1 to region 3 at step $t + 5$)

$(l_i^s)'$ and $\overline{\overline{\overline{l_i^s}}}$ attached both to membrane 3 at step $t+9$ to let $(Y^s)^{v\iota\iota\iota}$ move from region 1 to region 3.

So, we get the following result: at step $t + 5$ the object $(X^s)^{iv}$ is rewritten to # using rule 27.b or at step $t + 9$ the object $(Y^s)^{v\iota\iota\iota}$ is rewritten to # using rule 31.c.

Assumption (ii): In step $t + 2$ the object h_i^s does not move from region 1 to region 2 using a rule of group 7.*b*. This can only happen if $(l_i^s)'$ de-attaches, at step $t + 2$, from membrane 3 (using rule 7.c). But, in this case, the following condition cannot be fulfilled:

$(l_i^s)'$ and $\overline{\overline{\overline{l_i^s}}}$ both attached to membrane 3 at step $t+9$ (to let $(Y^s)^{v\iota\iota\iota}$ move from region 1 to region 3).

Therefore, at step $t + 9$, the object $(Y^s)^{v\iota\iota\iota}$ is rewritten to # using rule 31.c.

Assumption (iii): At step $t + 4$ the object $(l_i^s)'$ does not attach to membrane 3 using rule 9.a. In this case, at step $t + 5$, the object $(X^s)^{iv}$ cannot be moved from region 1 to region 3 and, hence, it is rewritten to # using rule 27.b.

Assumption (iv): Suppose at step $t + 6$ there is an object A_i in region 1. Then, in this step, using rule 10.a, A_i is moved inside region 3, where it is rewritten to # in the following step.

From the above description it follows that *all and only* the evolutions of Π that do not produce # in region 1 are the ones corresponding to correct simulations of derivations in G.

Moreover, as we have seen, when (one of) the rules that start a production simulation is applied (i.e., 14.a, 23, 24.a, 25.a), the objects $l_i' Y\ h_1 \cdots h_n$ or $(l_i^s)' Y^s X^s h_1 \cdots h_n$, for some $i \in Lab(P)$, and the objects corresponding to the current sentential form are the only ones present in region 1, while the object d is present in region 3 and region 2 and all the markings are empty.

Precisely:

There is a derivation in G producing the sentential form w if and only if there is an $i \in Lab(p)$ such that the two configurations of $\Pi\ [\ [\ ]^2 w' l_i' Y\ h_1 \cdots h_n\ [\ d]^3\ ]^1$ *and* $[\ [\ ]^2 w' (l_i^s)' Y^s X^s\ h_1 \cdots h_n\ [d\ ]^3\ ]^1$ *with* $\Psi_V(w) = \Psi_V(w')$ *are in* $C_R(\Pi, mp)$.

Also, from the constructive universality (see [11]), it is easy to show that it is not decidable whether or not an arbitrary programmed grammar with a.c. has a derivation of a sentential form with an arbitrary Parikh vector.

From this the Theorem follows. □

8 Conclusions and Open Problems

We have investigated a model of membrane systems with objects attached to both sides of the membranes and having operations that can rewrite floating objects and move objects between regions depending on the attached objects. We have proved that when the system works with free parallel evolution (i.e., allowing an arbitrary number of rules to be applied at each step) the reachability of a configuration or of a certain protein marking can be decided. We have also shown that when the system works with maximal parallel evolution (all rules that can be applied must be applied) the reachability of configurations becomes an undecidable property for the case of non-cooperative evolution rules and cooperative membrane rules. The property remains decidable, however, for systems using non-cooperative evolution rules and simple membrane rules and for systems using only membrane rules. An interesting problem remains open: the decidability of reachability in the case of systems using non-cooperative evolution rules, non-cooperative membrane rules and maximal-parallel evolution.

Several different directions may now be pursued.

Other bio-inspired operations may be introduced, such as *fission* and *fusion* of regions, all still dependent on the objects attached to the membranes, along the lines of the work found in [17]. In addition, the system could be analysed in the presence of timed rules, following the idea of time-independent P systems.

Another direction of research is the application of the model to simulate biological systems. To this end an implementation of a (more general) stochastic model has been created and can be found at [24]. The simulator has been used to model and simulate, among other things, a robust circadian clock and the receptor mediated G-protein cycle in yeast. For more details see [8].

References

[1] B. Alberts, *Essential Cell Biology. An Introduction to the Molecular Biology of the Cell*. Garland Publ. Inc., New York, London, 1998.

[2] N. Busi, R. Gorrieri, On the Computational Power of Brane Calculi. *Proceedings Third Workshop on Computational Methods in Systems Biology*. Edinburgh, 2005.

[3] R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, D. Sburlan, Membrane Systems with Marked Membranes. *Electronic Notes in Theoretical Computer Science, ENTCS*. To Appear.

[4] L. Cardelli, Brane Calculi. Interactions of Biological Membranes. *Proceedings Computational Methods in System Biology 2004* (V. Danos, V. Schächter, eds.), Lecture Notes in Computer Science, 3082, Springer-Verlag, Berlin, 2005.

[5] L. Cardelli, Gh. Păun, An Universality Result for a (Mem)Brane Calculus Based on Mate/Drip Operations. *Proceedings of the ESF Exploratory Workshop on Cellular Computing (Complexity Aspects)*, (M.A. Gutiérrez-Naranjo, Gh. Păun, M.J. Pérez-Jiménez, eds.), Fénix Ed., Seville, Spain. Also at `http://www.gcn.us.es/`.

[6] M. Cavaliere, Evolution-Communication P Systems. *Proceedings International Workshop Membrane Computing*, (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron eds.), Lecture Notes in Computer Science, 2597, Springer-Verlag, Berlin, 2003.

[7] M. Cavaliere, S. Sedwards, Membrane Systems with Peripheral Proteins: Transport and Evolution. *Proceedings MecBIC06. Electronic Notes in Theoretical Computer Science*, 171:2, pp. 37–53, 2007.

[8] M. Cavaliere, S. Sedwards, Modelling Cellular Processes Using Membrane Systems with Peripheral and Integral Proteins. *Proceedings International Conference on Computational Methods in Systems Biology, CMSB06*, Lecture Notes in Bioinformatics 4210, Springer, 2006.

[9] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds., *Applications of Membrane Computing*. Springer-Verlag, Berlin, 2006.

[10] V. Danos, S. Pradalier, Projective Brane Calculus. *Proceedings Computational Methods in System Biology 2004* (V. Danos, V. Schächter, eds.), Lecture Notes in Computer Science, 3082, Springer-Verlag, Berlin, 2005.

[11] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[12] D. Hauschildt, M. Jantzen, Petri Net Algorithms in the Theory of Matrix Grammars. *Acta Informatica*, 31, 1994.

[13] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[14] S.N. Krishna, Universality Results for P Systems based on Brane Calculi Operations. *Theoretical Computer Science*. To Appear.

[15] H. Lodish, A. Berk, S.L. Zipursky, P. Matsudaira, D. Baltimore, J. Darnell, *Molecular Cell Biology*, Freeman, Fifth Edition.

[16] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[17] A. Păun, B. Popa, P Systems with Proteins on Membranes and Membrane Division. *Proceedings Tenth International Conference in Developments in Language Theory, DLT06*, Lecture Notes in Computer Science, Springer-Verlag. To Appear.

[18] Gh. Păun, Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), pp. 108–143. First circulated as TUCS Research Report No 28, 1998.

[19] Gh. Păun, *Membrane Computing – An Introduction*. Springer-Verlag, Berlin, 2002.

[20] Gh. Păun, G. Rozenberg, A Guide to Membrane Computing. *Theoretical Computer Science*, 287-1, 2002.

[21] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[22] A. Salomaa, *Formal Languages*. Academic Press, New York, 1973.

[23] http://psystems.disco.unimib.it

[24] http://www.msr-unitn.unitn.it/downloads.php

Revised 30/9/09